Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of compiling a computer source program into a compressed compiler product containing differential names ~~in a compressed form~~, said method comprising:

receiving a source program written in a high-level programming language, the source program including one or more program symbols and non-program symbol information;

encoding a program symbol name to produce an encoded program symbol name, without changing the non-program symbol information;

~~generating~~ determining a differential name for the encoded program symbol name relative to a base symbol identifying a containing scope for the program symbol, wherein the containing scope is selected from a group consisting of: a namespace, a package, a module, a container object, and a function, and defines a context within which the differential name has an unambiguous meaning, and wherein the differential name ~~having a reduced size format as compared to~~ is formed at least in part by a sequence of characters constituting a subset of the encoded program symbol name; and

replacing the encoded program symbol name with the differential name to facilitate producing a compressed compiler product including the generated differential name.

2. (Canceled)

3. (Canceled)

4. (Currently Amended) A method as recited in claim 1, wherein said ~~generating comprises: identifying an~~ encoded program symbol name ~~within the compiler information that~~ is encoded in an extended format encoding. ~~;~~

~~determining a differential encoding for the encoded program symbol name relative to a base symbol for the program symbol, the differential encoding having a reduced size format as compared to the extended format; and~~

~~replacing the extended format encoding for the encoded program symbol name in the compiler information with the differential encoding.~~

5. (Currently Amended) A method as recited in claim 4, wherein said determining ~~generating~~ further comprises:

determining whether an augmented differential encoding is needed; and

if an augmented differential encoding is needed:

determining an encoded program symbol name identifier; and

attaching the encoded program symbol name identifier to the differential encoding.

6. (Previously Presented) A method as recited in claim 5, wherein the encoded program symbol name identifier is a base symbol identifier indicating a base symbol name associated with at least one of the encoded program symbol names.

7. (Original) A method as recited in claim 1, wherein the source program is written in a programming language selected from a group consisting of Ada, C++, Fortran, Pascal, and Java.

8. (Previously Presented) A method as recited in claim 1, wherein the compressed compiler product is an object code file.

9. (Previously Presented) A method as recited in claim 1, wherein the compressed compiler product contains debugger information.

10. (Currently Amended) A method of generating encoded program symbol names in an uncompressed form from compiled information containing differential names, ~~the encoded program symbol names being associated with compiler information,~~ said method comprising:

receiving a source program including one or more program symbol names;

determining whether any program symbol names are in a differential format;

for each program symbol name that is in a differential format:

extracting a differential program symbol name and a reference to a base symbol identifying a containing scope for the program symbol, wherein the containing scope is selected from a group consisting of: a namespace, a package, a module, a container object, and a function, and defines a context within which the differential name has an unambiguous meaning;

using the extracted reference to locate a non-differential name for the base symbol; and

~~decompressing~~ replacing the differential program symbol name with ~~based on~~ the non-differential name for the base symbol containing scope to obtain an encoded program symbol name in an uncompressed form, whereby a decompressed compiler product including the encoded program symbol name is produced.

11. (Canceled)

12. (Previously Presented) A method as recited in claim 10, wherein the base program symbol is a container of the program symbol that is represented by the compressed encoded program symbol name.

13. (Currently Amended) A ~~compilation~~ compiler system suitable for compilation and utilization of source programs, said compilation system comprising:

an enhanced compiler suitable for generation of enhanced compiler products, the enhanced compiler being operable to compile a source program having at least one program symbol name to produce the enhanced compiler products, the enhanced compiler products having a reduced size in comparison to compiler products produced by conventional compilers and including one or more differential names corresponding to the program symbol names; and

at least one enhanced non-compiler component operable to understand and utilize the enhanced compiler products.

14. (Original) A compiler system as recited in claim 13, wherein reduction of size of the enhanced compiler product is up to 40 percent of sizes of conventional compiler products produced by conventional compilers.

15. (Previously Presented) A compiler system as recited in claim 13, wherein the enhanced compiler product is a compiler related product selected from a group consisting of an object file, an executable file, and debugging information.

16. (Currently Amended) A computer readable medium including computer program code for compiling a computer source program into a compressed compiler product containing differential names ~~in a compressed form~~, said computer readable medium comprising:

computer program code for receiving a source program written in a high-level programming language, the source program including one or more program symbols and non-program symbol information;

computer program code for encoding a program symbol name to produce an encoded program symbol name, without changing the non-program symbol information;

computer program code for ~~generating~~ determining a differential name for the encoded program symbol name relative to a base symbol identifying a containing scope for the program symbol, wherein the containing scope is selected from a group consisting of: a namespace, a package, a module, a container object, and a function, and defines a context within which the differential name has an unambiguous meaning, and wherein the differential name ~~having a reduced-size format as compared to~~ is formed at least in part by a sequence of characters constituting a subset of the encoded program symbol name; and

computer program code for replacing the encoded program symbol name with the differential name to facilitate producing a compressed compiler product including the generated differential name.

17. (Canceled)

18. (Canceled)

19. (Currently Amended) A computer readable medium as recited in claim 16, wherein said ~~computer program code for generating comprises:~~

~~computer program code for identifying an~~ encoded program symbol name ~~within the compiler information that~~ is encoded in an extended format encoding;.

~~computer program code for determining a differential encoding for the encoded program symbol name relative to a base symbol for the program symbol, the differential encoding having a reduced-size format as compared to the extended format; and~~

~~computer program code for replacing the extended format encoding for the encoded program symbol name in the compiler information with the differential encoding.~~

20. (Previously Presented) A computer readable medium as recited in claim 16, wherein the compressed compiler related product is a compiler related product selected from a group consisting of an object file, executable file, and debugging information.

21. (Currently Amended) A computer readable medium including computer program code generating encoded program symbol names in an uncompressed form from compiled information containing differential names, the encoded program symbol names being associated with compiler information, said computer readable medium comprising:

computer program code for receiving a source program including one or more program symbol names;

computer program code for determining whether any program symbol names are in a differential format;

for each program symbol name that is in a differential format:

computer program code for extracting a differential program symbol name and a reference to a base symbol identifying a containing scope for the program symbol, wherein the containing scope is selected from a group consisting of: a namespace, a package, a module, a container object, and a function, and defines a context within which the differential name has an unambiguous meaning;

computer program code for using the extracted reference to locate a non-differential name for the base symbol; and

computer program code for ~~decompressing~~ replacing the differential program symbol name ~~based on~~ with the non-differential name for the base symbol to obtain an encoded program symbol name in an uncompressed form.

22. (Previously Presented) A method as recited in claim 1, wherein the base program symbol is a container object for the program symbol.